WHAT IS CLAIMED IS:

1. A method for removing dead code in code fragments of a program, comprising:

processing a first code fragment and storing first information generated during this processing indicative of whether an instruction for assigning a register in a first code fragment is possibly live;

processing a second code fragment and storing second information generated during this processing indicative of register usage;

at a time when the first and second code fragments are to be linked, determining, by use of the first and second stored information, if an instruction in the first code fragment that assigns a register is a dead instruction; and

responsive to determination that an instruction is a dead instruction, eliminating the dead instruction.

2. A method according to claim 1, wherein eliminating the dead instruction comprises overwriting the dead instruction with a NOP.

3. A method according to claim 1, wherein eliminating the dead instruction comprises compacting the surrounding instructions to delete the dead instruction.

4. A method according to claim 1, wherein:

the first information includes information associated with each exit from the first code fragment;

the second information includes information associated with each entry into the second code fragment;

the linking of the first and second code fragments links a particular exit from the first code fragment to a particular entry into the second code fragment;

the step of determining uses the first information associated with the particular exit and the second information associated with the particular entry.

5. A method according to claim 4, wherein the first information associated with each exit includes a pointer to each instruction for assigning a register that is possibly live for that exit.

6. A method according to claim 5, wherein the first information associated with each exit further includes a first register mask, the first register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in an instruction pointed to by a pointer in the first information associated with that exit.

7. A method according to claim 6, wherein the second information associated with each entry includes a second register mask, the second register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in the second fragment before being read.

8. A method according to claim 7, where said determining step comprises comparing corresponding positions of the first and second register masks, wherein said eliminating step includes eliminating an instruction for assigning a register in the first code fragment if the positions corresponding to the register in the first and second register masks are both set.

9. A method according to claim 8, wherein said eliminating step further comprises determining which instruction to overwrite with reference to the pointers in first information.

10. A method according to claim 4, wherein the first information associated with each exit is stored in an epilog associated with that exit, and the second information associated with each entry is stored in a prolog associated with that entry.

11. A computer readable comprising instructions for removing dead code in code fragments of a program, the instructions configured to:

process a first code fragment and store first information generated during this processing indicative of whether an instruction for assigning a register in a first code fragment is possibly live;

process a second code fragment and store second information generated during this processing indicative of register usage;

at a time when the first and second code fragments are to be linked, determine, by use of the first and second stored information, if an instruction in the first code fragment that assigns a register is a dead instruction; and

responsive to determination that an instruction is a dead instruction, eliminate the dead instruction.

12. A computer readable medium according to claim 11, wherein eliminating the dead instruction comprises overwriting the dead instruction with a NOP.

13. A computer readable medium according to claim 11, wherein eliminating the dead instruction comprises compacting the surrounding instructions to delete the dead instruction.

14. A computer readable medium according to claim 11 wherein:

the first information includes information associated with each exit from the first code fragment;

the second information includes information associated with each entry into the second code fragment;

the linking of the first and second code fragments links a particular exit from the first code fragment to a particular entry into the second code fragment;

the step of determining uses the first information associated with the particular exit and the second information associated with the particular entry.

15. A computer readable medium according to claim 14, wherein the first information associated with each exit includes a pointer to each instruction for assigning a register that is possibly live for that exit.

16. A computer readable medium according to claim 15, wherein the first information associated with each exit further includes a first register mask, the first register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in an instruction pointed to by a pointer in the first information associated with that exit.

17. A computer readable medium according to claim 16, wherein the second information associated with each entry includes a second register mask, the second register mask having a plurality of positions, each position corresponding to a respective register, wherein a bit at a position is set if the respective register is assigned in the second fragment before being read.

18. A computer readable medium according to claim 17, where said determining step comprises comparing corresponding positions of the first and second register masks, wherein said eliminating step includes eliminating an instruction for assigning a register in the first code fragment if the positions corresponding to the register in the first and second register masks are both set.

19. A computer readable medium according to claim 18, wherein said eliminating step further comprises determining which instruction to overwrite with reference to the pointers in first information.

20. A computer readable medium according to claim 14, wherein the first information associated with each exit is stored in an epilog associated with that exit, and the second information associated with each entry is stored in a prolog associated with that entry.